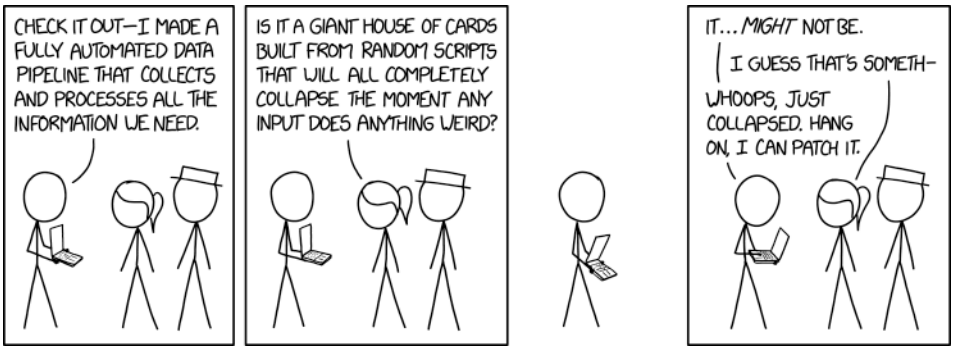
Program Assignment
CPE 357
Systems Programming

CAL POLY
SAN LUIS OBISPO

Assignment 3



Exercise:

Write a “findfile” program.
The findfile program will let you enter its own shell:
findstuff$
Now you should be able to enter commands and react to them:
Important: I leave the exact formatting and text up to you, but hold up to the premise

- find <filename> -flag .. tries to find this specific file. For that, your program fork()’s a new child process which is doing that. Assign a serial exclusive number to this and each child, you will need it later with the “kill” command.
  Once done, it interrupts* the scanf from the parent process and prints its result. The result should be something like >nothing found< in case nothing was found, but if, then print the file’s name and its directory. Print several lines if several files with the same name were found.

  e.g.:
  findstuff$ find test.c
  text.c bin/usr/desktop

- flags:
  There are different flags:
  No flag … Search in the current directory.
  -s … search in the current directory and all sub-directories.
  -f … search in all directories

- quit or q .. quits the program and all child processes immediatelly.

Have no more than 10 childs at a time. Report if the user attempts to exceed that limit and nicely print, that the limit is reached.

*interrupt

How to interrupt while the parent is in scanf-mode waiting for another input?
Just interrupt with printf, the scanf will still be active later on.
Redirecting stdin is not a good idea, because once redirected, it does not (at least not that easy and not on all systems let you get data from the keyboard again.

What for? Why child processes, why not simply use one process?

Bevcause seaching for one file across the whole harddrive can take minutes. During this time, the user should be able to search for other files. This is why we use child processes for doing so.
Also:
Learning pipe(), dup2() and how to redirect stdin, lots of programing exercise due to string handling, nice use of signals and on top a useful program.

How we test

We send a couple of find request on the way on a big HD. So they will need some time (>3sec). Then we check if everything is nicely reporting back.

Submission:

Submit the source code file(s) and the executables: MYNAME_findstuff_ass4**.zip**

FAQ (this section will be extended in the next days)

>What happens with the child when its done? E.g. finding the file(s).

1. Report its findings by printing the result.
2. It should end. To avoid a zombie, wait for that specific child with waitpid() (https://linux.die.net/man/2/waitpid) which will be discussed in the Wednesday's video. How does the parent know to wait for that child? Signal! And send its PID into a pipe or shared mem!

You need:

waitpid()
opendir()
closedir()
readdir()
pipe()
dup()
dup2()

A good Idea how to start:

- Start using your lab code pipelines. It already has the redirecting.
- Write a parsing function, to extract the arguments.
- Write a recursive function to get all entries of a directory, something like:
  (pseudocode)

```
void findstuff(char* filename_to_be_found, char *startdirectory,char *result,int
search_in_all_subdirectories)
{
opendir(startdirectory)
readdir();
if (entryname == filename_to_be_found) write to result;
if(entry_is_subdirectory && search_in_all_subdirectories)
        findstuff(filename_to_be_found, startdirectory += '/' + entryname, result,
        search_in_all_subdirectories)

closedir();
}
```

- Make sure not having zombies.

And finally, find the bug!: